REMARKS

This amendment is submitted in response to the Office Action dated November 17, 2005. Reconsideration and allowance of claims is requested.

In this Office Action, claims 1-20 were examined, and all rejected under 35 U.S.C. 102(b) as anticipated by Arimilli (U.S. 6,463,507). This rejection is respectfully traversed.

The present invention, as now more clearly claimed, is directed to an environment in which a plurality of execution units are simultaneously processing threads comprising sequences of instructions that operate on data. The operations require frequent access to constants which are supplied to data caches associated with each of the execution units. The claimed organization is adopted because the different threads being executed may require access to the same constant. Therefore, an execution unit would frequently have it's operations stalled under a number of circumstances, for example, where another execution unit had access to the constant, or to insure that a constant is not inadvertently modified before being read by the execution unit.

In the prior art, as explained at application paragraph [0004], the controller would need to determine that each execution unit is idle before writing a constant to storage, potentially blocking a unit requiring access to the constant and significantly reducing the throughput of the processing system. When many execution units are utilized, one or more execution units may be idle for many clock cycles while other processing units complete processing before also becoming idle.

To solve this problem, the invention claimed herein covers a scheme (an example of which is shown in figure 2 and a description of which is found beginning at paragraph [0021]) wherein each execution unit is able to process a thread using the same or a different value for a constant because a dedicated level 1 cache is associated therewith, where each such cache is capable of storing a different version of the constant. By adopting this approach, the execution of each thread proceeds independently of the other threads. As noted, each execution unit may process one or several threads, with either the same or different constants being made available in the different level 1 caches and each constant is identified in part by an associated version tag. As an execution unit becomes inactive, the constant is moved to the level 2 cache and marked (by the version tag) as being associated with an older active thread; the constants found in the level 2 cache are

coherent with some of the level 1 caches that are still executing the older threads but not other caches which are associated with execution units executing a newer or different thread.

The system further allows for locking each level 1 cache from being overwritten prior to either the invalidating the level 1 cache or moving the constant to a level 2 cache to solve the problem described above of a constant being changed before it is used in all threads. The system also provides for moving a constant from one level 1 cache to another level 1 cache if the execution of threads by different execution units requires such access to constants.

None of these features are taught or suggested in the Arimilli reference cited by the Examiner which simply provides for a level 1 cache 200 and a level 2 cache 202 with every data access being provided simultaneously to both caches and the data type being taken from whichever cache immediately provides access to the data. No disclosure or discussion of a plurality of executions units, each with an associated level 1 cache, or the association of a version tag with each stored constant is provided. There is no discussion of moving of constants associated with the oldest thread under execution being moved to the level 2 cache, with the controller updating the associated version tag to reflect such movement, or of locking the caches during execution.

In view of these fundamental distinctions, among others, between the invention disclosed and claimed in the present application and the reference cited by the Examiner, reconsideration and allowance of the claims is respectfully requested.

Respectfully submitted,

James A Sheridan

James A. Sheridan
Registration No. 25,435
PATTERSON & SHERIDAN, L.L.P.
3040 Post Oak Blvd. Suite 1500
Houston, TX 77056
Telephone: (713) 623-4844
Facsimile: (713) 623-4846
Attorney for Applicants